



The Smarter Way to Handle Guest Images

Understanding libguestfs, a library to access and modify VM images.

Most administrators might have experienced the ease of using Sysprep for the automated deployment of Windows-based machines, especially virtual desktops. Automated deployment of virtual desktops and servers in batches is inevitable in an IT infrastructure that has virtualisation. There are certain things that are unique in every guest, like the IP address, hostname, licences, certain configuration file parameters, etc, which need to be updated at the time of deployment.

I had been thinking a lot on what could be the equivalent on the Linux side when it comes to deploying Linux-based virtual machines.

Around this time, I upgraded one of my boxes to Fedora 12 and learned about the availability of libguestfs and guestfish. The interactivity of the utility may not be as robust as Sysprep, and libguestfs alone may not be equivalent to Sysprep in all aspects, but it bridges a big gap in the automated deployment of virtual machines.

Deploying virtual machines from a template has always been a challenge due to the diversity of requirements and use-cases for different machines. The traditional way is to deploy the virtual machine from a predefined general template and customise it later for specific requirements. This is absolutely non-scalable and is a huge burden on the administrator.

libguestfs helps overcome this challenge. Since Anaconda installation is not done while virtual machines are deployed from pre-defined templates, using *kickstart %post* is not an option. libguestfs can be used to automatically make these required modifications inside a guest filesystem before the guest is deployed.

libguestfs is a library used to inspect, modify and manipulate virtual machine disk images. It supports almost all virtual disk formats like the raw image, vmdk (VMware image format), qcow2, LVM block devices, etc, and filesystems like ext2, ext3, ext4, btrfs, FAT, NTFS, etc.

The libguestfs tools

libguestfs provides some tools to manipulate the images efficiently. These are provided through the *libguestfs-tools* package in Fedora and can be used on the guest images non-interactively.

- *virt-rescue*: Launches a shell in the guest image to do rescue operations like editing boot configuration files, modifying initrd images, etc.
- *virt-edit*: Opens the specified file inside the guest image in your favourite editor.
- *virt-inspector*: Gets basic details about the guest from its image, like the kernel version, modules loaded, mount points, details of applications installed in the guest, etc.

Other tools are *virt-cat*, *virt-ls*, *virt-rescue*, *virt-df*, *virt-tar* and *virt-win-reg*—I'm assuming readers will find the names self-explanatory. *virt-win-reg* is used to display Registry Entries from a Windows guest image.

The syntax of how to use *virt-cat* is as follows:

```
# virt-cat <guest image or block device> <file name>
```

For example:

```
# virt-cat f12.img /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.
localhost4
::1        localhost localhost.localdomain localhost6 localhost6.
localhost6
```

Checking the man pages for the respective tools will provide more details on how to use them.

There are certain tools developed with the help of libguestfs. *virt-v2v* is one of them, and is used to migrate virtual machines running on one platform to another one (for example, virtual machines running on VMware to KVM). libguestfs is used extensively by this tool to modify the filesystem images to run appropriately on the target platform.

libguestfs can be used to inspect a guest filesystem while in offline mode and collect data for administrative purposes.

guestfish

guestfish uses libguestfs to launch a shell on the guest filesystem to help us modify the image efficiently, offline or online. It's recommended not to invoke it on running guests and make

modifications, due to the possibility of filesystem corruption.

You can invoke guestfish on online guests in read-only mode to view details. You can call it from a shell script, invoked from the command line directly, without launching an interactive shell to execute commands inside the guest image. You can also use it to launch an interactive shell on the libguestfs filesystem and execute commands from the shell. Examples of each method are available in its respective man page.

You need to use the *-i* switch to execute guestfish to launch an interactive shell:

```
# guestfish -i f12.img
```

```
Welcome to guestfish, the libguestfs filesystem interactive shell for
editing virtual machine filesystems.
```

```
Type: 'help' for help with commands
      'quit' to quit the shell
```

```
><fs>
```

Typing help here will display all the available commands inside guestfish and running the help *<command>* will give more details about the specific command.

Troubleshooting

You can also use libguestfs to make modifications inside a guest image for troubleshooting purposes. This makes troubleshooting on a broken guest very easy. An administrator does not need to go into rescue mode to reinstall Grub, make modifications to boot loader configuration files, etc. Let us look at the simple example of a kernel upgrade rendering the system non-bootable. Reverting to the old well-known working kernel is as simple as running the following command:

```
# guestfish -i <guest image/ blockdevice>
```

...and editing the */boot/grub/grub.conf* file to change the default kernel to boot.

In the same manner, we can reinstall the boot loader on the guest image if it somehow gets corrupted. There is no need to follow the old way of looking for a bootable CD and going to rescue mode to reinstall it. As a matter of fact, this way of recovering a broken production virtual machine becomes much faster compared to recovering a broken physical system. Let us have a look at how we can reinstall Grub on a Fedora 12 guest image.

Use *virt-df* on the image to view the filesystem details of the guest:

```
# virt-df f12.img
```

Filesystem	1K-blocks	Used	Available	Use%
f12.img:/dev/VolGroup/lv_root	6821604	1346828	5128256	24.8%
f12.img:/dev/sda1	198337	21435	166662	16.0%

This shows that the / filesystem of the guest is on /dev/VolGroup/lv_root and /boot is on /dev/sda1.

Next, launch an interactive guestfish filesystem shell on the image by mounting both / and /boot filesystems, appropriately.

```
# guestfish -a f12.img -m /dev/mapper/VolGroup-lv_root -m /dev/sda1:/boot
```

```
Welcome to guestfish, the libguestfs filesystem interactive shell for
editing virtual machine filesystems.
```

```
Type: 'help' for help with commands
      'quit' to quit the shell
```

```
><fs> df
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup-lv_root
                6821604      1346828    5128256    21% /sysroot
/dev/sda1        198337        21435     166662    12% /sysroot/boot

><fs> grub-install /dev/sda
```

In the above command, the image or the block device on which the guest is installed should follow the *-a* switch. Each block device that needs to be mounted from the guest should follow the *-m* switch in the format of the 'block device:mount point'. We need to mount /boot since the grub-install command installs stage files in the /boot/grub/ directory.

Template

Most virtualisation technologies support deploying virtual desktops from templates using the copy-on-write (COW) method. libguestfs will be very useful in making modifications on the template, running security updates on the template, etc.

For example, let's assume 200 virtual desktops are deployed from a template using the copy-on-write method. While using this method, changes specific to a virtual desktop from the actual template will be stored on a different file. The template will be used in conjunction with the changes on the file to form the guest filesystem. If the kernel needs to be updated on these 200 virtual desktops, it's as simple as taking a copy of the template, using libguestfs to update the kernel on the copy of the template, and changing the template for the guest to use to the new template with the updated kernel. On the next reboot, all the 200 virtual desktops should have the updated kernel. Likewise, making batch changes to configuration files is also very easy, using this method.

Windows support

libguestfs has minimal support when it comes to handling Windows-based images. FAT and NTFS filesystems are fully supported. You can use *virt-win-reg* to view the values from the Windows registry. For example, I used

the following command to get the product name from a Windows 2003 image:

```
# virt-win-reg windows-2003.img '\HKEY_LOCAL_MACHINE\Software\Microsoft\
Windows NT\CurrentVersion' ProductName
Microsoft Windows Server 2003
```

Partial back-up

Taking a full back-up of a virtual machine is very easy. Just backing up the whole image or taking lvm snapshots or 'dd-ing' block devices will suffice. On physical systems, taking a partial back-up (e.g., taking a back-up of Apache configuration files only) usually involves copying content from inside the virtual machine over the network, using rsync, scp or other back-up utilities. Libguestfs can be used to take a partial back-up of the guest from the host without logging into the guest. However, it needs to be noted that, for a reliable back-up, I strongly recommend that you do not use libguestfs. A back-up agent should be installed inside the guest, like we do with ordinary machines.


Cloning a virtual machine

Cloning a virtual machine is usually done by copying the image of a virtual machine to a different location and replacing details in the image that are unique to a virtual machine, like the hostname, IP address, MAC address, etc. You can use libguestfs to serve this purpose, whereas these unique details will either be replaced with actual details provided by the user or replaced by a placeholder.

Areas for improvement in libguestfs

One thing that I expected to see in libguestfs was support for Solaris filesystems like UFS, ZFS, etc. Unfortunately, it's not available as of now. (Note, that this isn't a libguestfs drawback, *per se*. If Linux has support for them, libguestfs will support them. Also, there are some licensing restrictions when it comes to supporting ZFS.)

Caution

One thing that needs to be kept in mind while dealing with libguestfs is not to make modifications when using it on running guests. There is a strong possibility that this will corrupt the guest filesystem beyond recovery. Future versions of libguestfs will introduce thorough locking to prevent people from doing this. At this moment, it's recommended to always connect to running guests using the *—ro* switch to guestfish, to prevent modifications.  **END**

Resources

- libguestfs home: www.libguestfs.org
- Richard W M Jones' Blog: rwmj.wordpress.com

By: Sadique Puthen

The author is an open source enthusiast with deep interest in Linux and virtualisation technologies. He is currently working as a senior technical support engineer at Red Hat, and can be reached at sputhenp@redhat.com